C# Programlama

Temel C# Sözdizimi

Emir Öztürk

Uygulama Girişi

- Konsol uygulaması
- Eski tip
- Yeni tip
- Parametre alınmak istendiğinde eski tip

```
using System;
namespace Lecture
{
    class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

C# Program.cs

```
1   Console.WriteLine("Hello World!");
```

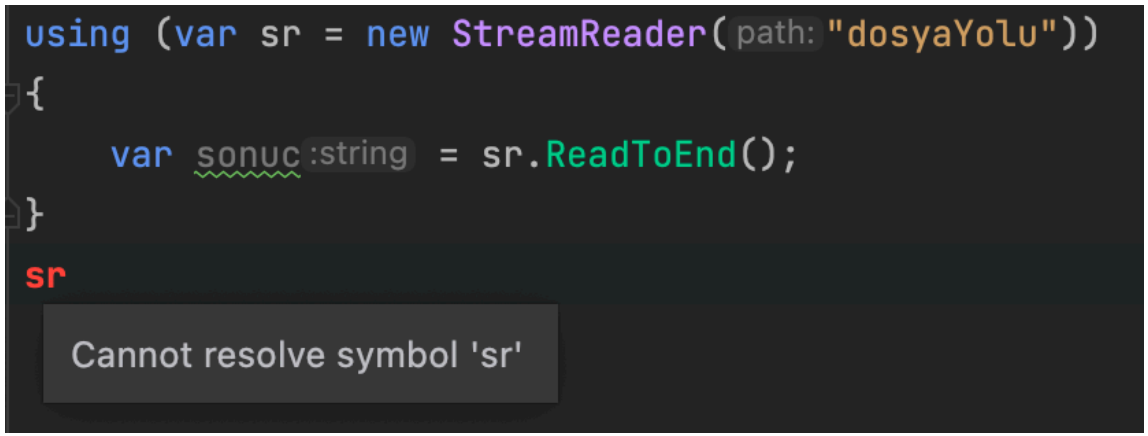
Using

- #include, import
- Paket veya isimuzayı
- Kullanılmadığı durumda Gri
- Genel isimuzaylarının yazılması gerekmiyor

```
using System.Collections;
using System.IO;
using System.Formats;
Console.WriteLine("Hello World!");
var sr = new StreamReader(path: "");
var ba = new BitArray(values: new []{true,false});
```

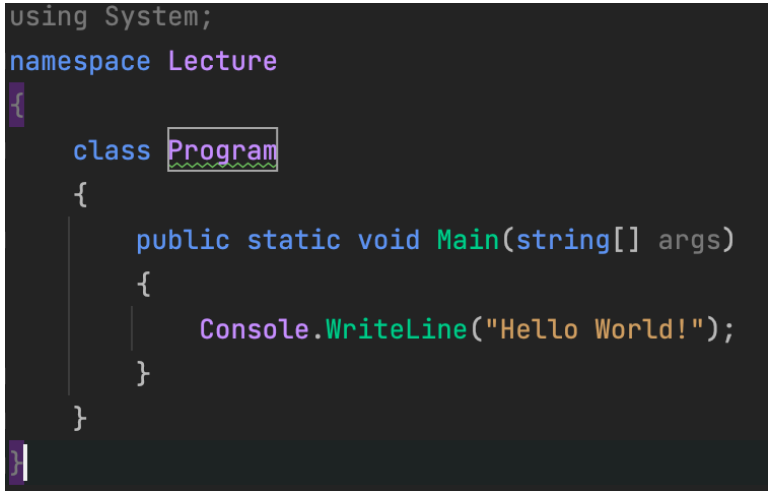
Using

- Kod içerisinde çöp toplayıcısına yardım için kullanılabilir
- Using dışına çıkıldığında değişkene erişilemez



Main

- Yeni konsol uygulamalarında gerekmiyor
- Dışarıdan parametre alınması istendiğinde kullanılabilir
- Statik
- Void
- String args[]



Değişken Türleri

- bool
- byte (sbyte)
- short (ushort)
- int (uint)
- long (ulong)
- float
- double
- char
- decimal

Operatörler

Operatör	İşlem
+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme
%	Mod Alma

Operatör	İşlem
==	Eşit mi
>	Büyük mü
<	Küçük mü
>=	Büyük eşit mi
<=	Küçük eşit mi
!=	Farklı mı

Operatörler

X	Y	Or	And
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE

Operatör	İşlem
++	Bir arttır
--	Bir eksilt
+	İşareti sabit bırak
-	- işarete çevir
!	Değilini al

Operatörler

Operatör	İşlem
?:	if-else
-	İkili komplement
&	And
|	Or
^	Xor
<<	Left shift
>>	Right shift

Operatör	İşlem
+=	Topla eşitle
-=	Çıkart eşitle
***=**	Çarp eşitle
/=	Böl eşitle
%=	Mod al eşitle
&=	And al eşitle
|=	Or al eşitle
^=	Xor al eşitle
<<=	Sola kaydır eşitle
>>=	Sağa kaydır eşitle
=>	Lambda operatörü

Koşullar

- if
  - else if
  - else
- switch
  - case
  - default
- Yeni Switch

```
var degisken = Convert.ToInt32(Console.ReadLine());

if(degisken == 1)Console.WriteLine("Bir");
else if(degisken==2)Console.WriteLine("İki");
else Console.WriteLine("Yanlış");
```

```
var degisken = Convert.ToInt32(Console.ReadLine());

switch (degisken)
{
    case 1:
        Console.WriteLine("Bir");
        break;
    case 2:
        Console.WriteLine("İki");
        break;
    default:
        Console.WriteLine("Yanlış");
        break;
}
```

```
var degisken = Convert.ToInt32(Console.ReadLine());

var sonuc :string = degisken switch
{
    1 => "Bir",
    2 => "İki",
    _ => "Yanlış"
};
Console.WriteLine(sonuc);
```

Diziler

- Tanım

```
int[] dizi = new int[10];
var dizi2 = new int[10];
var dizi3 = new int [10]{1,2,3,4,5,6,7,8,9,10 };
var dizi4 = new int []{1,2,3,4,5,6,7,8,9,10 };
var dizi5 = new []{1,2,3,4,5,6,7,8,9,10 };
int[] dizi6 = {1,2,3,4,5,6,7,8,9,10};
```

- Erişim

```
Console.WriteLine(dizi[0]);

dizi[0] = Int32.MaxValue;
```

- Boyut

```
var uzunluk:int = dizi6.Length;
```

Döngüler

- For

```
for(int i=0;i<dizi.Length;i++)
    Console.WriteLine(dizi[i]);
```

- Foreach

```
foreach(var eleman:int in dizi)
    Console.WriteLine(eleman);
```

- While

```
int x = 0;
while (x < dizi.Length)
    Console.WriteLine(dizi[x++]);
```

- Do-While

```
int y = 0;
do
{
    Console.WriteLine(dizi[y++]);
} while (y < dizi.Length);
```

Try-Catch-Finally

- Hata yakalama
- try - çalıştırma
- catch - hata
- finally - her durum sonunda

Performansa etkisi

Defansif Programlama

```
int satirSayisi;
try
{
    var satirlar :string[] = File.ReadAllLines( path: "Yol");
    satirSayisi = satirlar.Length;
}
catch (IOException ex)
{
    Console.WriteLine(ex.Message);
}
finally
{
    satirSayisi = -1;
}
```

Yapılar

- struct kelimesi
- Örneği alınabilir / değer türünde kullanılabilir
- Yapıcı ve metot içerebilir

```
struct Kisi
{
    public string Ad;
    public string Soyad;
}
```

```
struct YapicisiOlanKisi
{
    private string Ad;
    private string Soyad;

    public YapicisiOlanKisi(string ad, string soyad)
    {
        Ad = ad;
        Soyad = soyad;
    }

    public string GetAd()
    {
        return Ad;
    }

    public string GetSoyad()
    {
        return Soyad;
    }
}
```

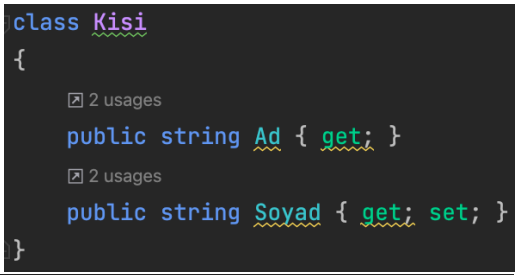
Yapılar

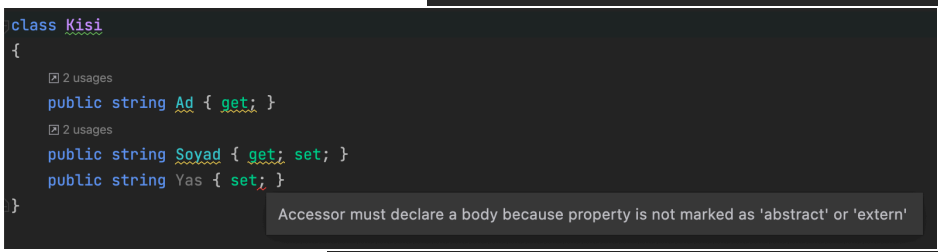
- Yapıcısı yoksa public tanımında alanlar atanabilir
- private tanımlanmış alanlar yapıcı ile tanımlanabilir
- Yapıcı olmadığı takdirde {} sentaksı ile ilk değer ataması yapılabilir

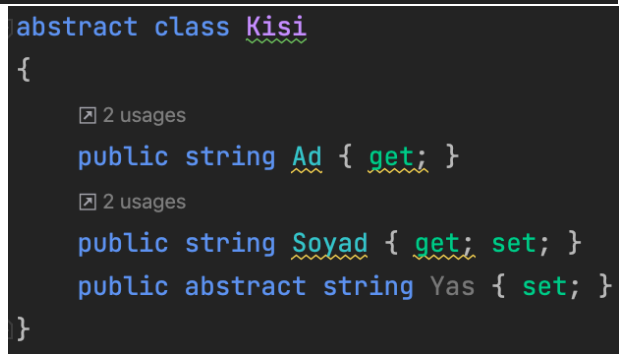
```
class Program
{
    public static void Main(string[] args)
    {
        var yeniKisi = new Kisi();
        yeniKisi.Ad = "Emir";
        yeniKisi.Soyad = "Öztürk";

        var ikinciKisi = new Kisi{ Ad = "Emir", Soyad = "Öztürk" };

        var yapicisiOlanKisi = new YapicisiOlanKisi(ad: "Emir", soyad: "Öztürk");
    }
}
```

Alanlar

- Getter setter yazmak yerine kullanılabilir
- Yalnızca get edilebilir
- Yalnızca set edilebilir







Sınıflar

- class kelimesi
- Her zaman heap’te
- Örnek alınmalı (new)

```
1 usage
class Kisi
{
    1 usage
    public string Ad { get; set; }
    1 usage
    public string Soyad { get; set; }
    1 usage
    public int Yas { get; set; }

    1 usage
    public Kisi(string ad, string soyad, int yas)
    {
        Ad = ad;
        Soyad = soyad;
        Yas = yas;
    }
}

class Program
{
    public static void Main(string[] args)
    {
        var kisi = new Kisi(ad: "Emir", soyad: "Öztürk", yas: 42);
    }
}
```

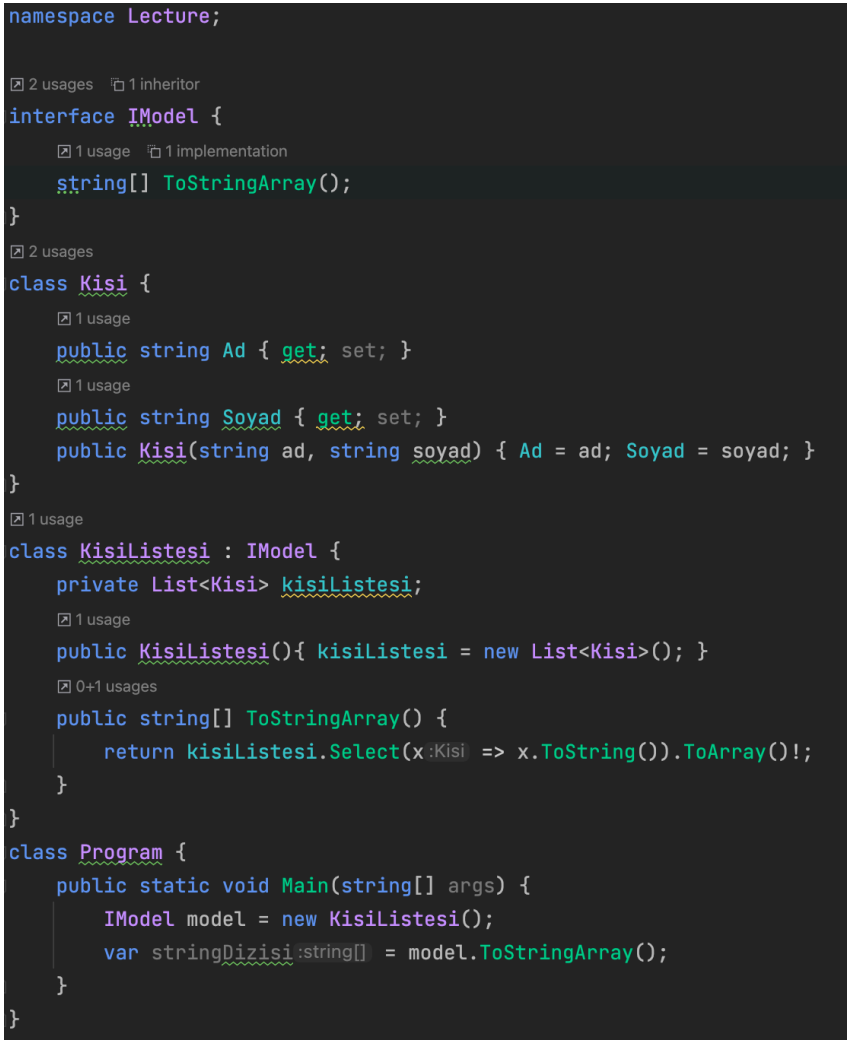
New

- Türün örneği
- Heap
- Çöp toplayıcı
- Temel türler / değer türünde struct

```
1 usage
class Kisi
{
    1 usage
    public string Ad { get; set; }
    1 usage
    public string Soyad { get; set; }
    1 usage
    public int Yas { get; set; }

    1 usage
    public Kisi(string ad, string soyad, int yas)
    {
        Ad = ad;
        Soyad = soyad;
        Yas = yas;
    }
}

class Program
{
    public static void Main(string[] args)
    {
        var kisi = new Kisi(ad: "Emir", soyad: "Öztürk", yas: 42);
    }
}
```

Arayüzler

- Şablonlar
- Sınıflar tarafından içerilir
- İçeren sınıfların implementasyonu



```
namespace Lecture;

2 usages  1 inheritor
interface IModel {
    1 usage  1 implementation
    string[] ToStringArray();
}
2 usages
class Kisi {
    1 usage
    public string Ad { get; set; }
    1 usage
    public string Soyad { get; set; }
    public Kisi(string ad, string soyad) { Ad = ad; Soyad = soyad; }
}
1 usage
class KisiListesi : IModel {
    private List<Kisi> kisiListesi;
    1 usage
    public KisiListesi(){ kisiListesi = new List<Kisi>(); }
    0+1 usages
    public string[] ToStringArray() {
        return kisiListesi.Select(x :Kisi => x.ToString()).ToArray()!;
    }
}
class Program {
    public static void Main(string[] args) {
        IModel model = new KisiListesi();
        var stringDizisi :string[] = model.ToStringArray();
    }
}
```